

INSIDE DOS

Tips & techniques for MS-DOS & PC-DOS

Strategies for organizing directories

The directories on a PC are in many ways like file folders—and we all know how personalized filing systems can be. But unlike file folders, you can create hierarchical relationships among your directories. *Inside DOS* subscriber Jim Walsh recently asked if we had any tips for taking better advantage of DOS' directory capabilities. Unfortunately, DOS users agree on few absolute rules for organizing directories. However, we can outline some typical methods for organizing directories on a PC and consider their advantages and pitfalls. In this article, we'll explore the following rules of thumb for creating and organizing directories:

- placing only essential files in your root directory
- limiting the files in any one directory
- separating program files from data files
- categorizing files by subject or application—or both—to decide in which directory they belong

As we'll see, the first three concepts are fairly easy to follow, but the more abstract concept of categorizing your files can be a bit trickier. Let's start by looking at the most fundamental directory on your hard disk: the root directory.

The root directory

DOS purists insist that the root directory contain only files your PC needs to boot up. According to them, you should place only the AUTOEXEC.BAT file, the CONFIG.SYS file, and COMMAND.COM—DOS' command interpreter—in your root directory. In practice, many DOS users also keep favorite batch files, utilities, and even document templates in the root directory. In theory, you could have a total of 512 files and subdirectories in the root directory. (The sidebar "A Directory of Subdirectories" on page 12 shows you several ways to list the subdirectories on your hard disk.)

Of course, if you place the directories that contain batch files or utilities on your path, you'll still be able to run them from your root directory. For example, you can keep all your batch files in the C:\BATCH directory. By including the C:\BATCH directory in AUTOEXEC.BAT's PATH command, you can run batch files from any

directory. As you can see below, you could run the batch file C:\BATCH\COLOR.BAT from the root directory:

```
C:\>color
```

Since C:\BATCH is on your path, DOS will look there for batch files, command files, and executable program files (that is, files with the BAT, COM, or EXE extensions). Consequently, it will find and run COLOR.BAT.

Directory overload

Although you should place only your most important files in the root directory, you can be more liberal with the directories that branch off from the root directory. (Technically, any directory that branches off from the root directory is a subdirectory. However, these subdirectories are commonly called "directories" as well.)

While DOS 5 sets no limits on the number of files you can place in a subdirectory, you'll probably end up setting a practical limit. For example, you might limit directories by the number of files displayed on one screen. Since monitors display 25 lines per screen, you might limit your directories to 20 files so they'll fit on one

Continued on page 10

IN THIS ISSUE

- Strategies for organizing directories 1
- Entering ASCII codes with the DOS 5 Editor 2
- Introducing The Cobb Group Online 3
- Online glossary 3
- Selecting larger display type with ANSI codes 4
- Choosing display modes with SCREEN.BAT 6
- Van Wolverton: Taking care of the environment 7
- A roundup of directory commands 11
- A directory of subdirectories 12

INSIDE DOS™

Inside DOS (ISSN 1049-5320) is published monthly by The Cobb Group.

Prices Domestic \$49/yr. (\$6.00 each)
Outside US \$69/yr. (\$8.50 each)

Phone Toll free (800) 223-8720
Local (502) 491-1900
FAX (502) 491-8050

Address You may address tips, special requests,
and other correspondence to:

The Editor, *Inside DOS*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

For subscriptions, fulfillment questions, and requests for
bulk orders, address your letters to:

Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

Back Issues To order back issues, call Customer Relations at (800)
223-8720. Back issues cost \$6.00 each, and you can
pay with MasterCard, VISA, Discover, or American
Express, or we can bill you. Please identify the issue
you want by the month and year it was published.
Customer Relations can also provide you with an issue-
by-issue listing of all the articles that have appeared in
Inside DOS.

Postmaster Second class postage is paid in Louisville, KY. Send
address changes to:

Inside DOS
P.O.Box 35160
Louisville, KY 40232

Copyright Copyright © 1992, The Cobb Group. All rights are
reserved. *Inside DOS* is an independently produced
publication of The Cobb Group. Readers who wish to
share tips should write to The Editor, *Inside DOS*, 9420
Bunsen Parkway, Suite 300, Louisville, KY 40220. The
Cobb Group reserves the right, with respect to
submissions, to revise, republish, and authorize its
readers to use the tips submitted for both personal and
commercial use. The Cobb Group, its logo, and the
Satisfaction Guaranteed statement and seal are
registered trademarks of The Cobb Group. *Inside DOS*
is a trademark of The Cobb Group. Microsoft is a
registered trademark of Microsoft Corporation. IBM is a
registered trademark of International Business Machines
Corporation.

Staff Editor-in-Chief Suzanne Thornberry
Contributing Editor Van Wolverton
Editing Clyde Zellers
Polly Blakemore
Cecilia Crosby-Lampkin
Production Margueriete Stith
Design Karl Feige
Publications Manager Elayne Noltemeyer
Circulation Manager Ron Schweinhart
Publications Director Linda Baughman
Editorial Director Joe Pierce
Publisher Douglas Cobb

LETTERS

Entering ASCII codes with the DOS 5 Editor

Since I installed DOS 5, I've abandoned EDLIN in favor of the new Editor, EDIT.COM. I use the Editor regularly to edit and print short ASCII files. However, I've found one thing that I could do in EDLIN that I can't do in the DOS 5 Editor: entering the form feed character. Do you know how to create this character with the new Editor?

Robert D. Marshall
Orlando, Florida

We encountered the same problem when we switched to the DOS 5 Editor. Todd Martin of Microsoft Technical Support explained that the trick is to press [Ctrl]P before you enter the ASCII code for the special character you want to place in your file. For example, to create the form feed character, you press [Ctrl]P, then hold down the [Alt] key while you type the number 12.

You'll need to use the [Ctrl]P method to enter several special characters that are sometimes used in batch files. Interestingly, you can use an alternate method to create most of the special characters—except the form feed. In addition to the [Alt]-number method for entering ASCII codes, you can instead use the [Ctrl]-letter combination assigned to the special character. For example, you can enter the special character that creates your computer's beep (or "bell") by pressing [Ctrl]P, then [Ctrl]G. Or, you can press [Ctrl]P, then [Alt]7. Table A shows both the numeric and letter codes for the special characters you can use in batch files.

Table A

Function	[Ctrl]-letter	[Alt]-number
Bell	[Ctrl]G	[Alt]7
Backspace	[Ctrl]H	[Alt]8
Horizontal tab	[Ctrl]I	[Alt]9
Linefeed	[Ctrl]J	[Alt]10
Vertical tab	[Ctrl]K	[Alt]11
Form feed	none	[Alt]12
Enter (Return)	[Ctrl]M	[Alt]13
Escape character	[Ctrl][[Alt]27

When you enter these characters, the DOS 5 Editor will display a special symbol for each one. For example, when you press [Ctrl]P[Ctrl]G, DOS will display as a small diamond the character that sounds the bell. ■

Introducing The Cobb Group Online

Beginning this month, we have a new service to offer you. The Cobb Group Online, available through ZiffNet, is an easily accessible source of online technical support devoted to the needs of DOS users. Created by Ziff Communications, ZiffNet is a comprehensive online information service that brings together the best in downloadable freeware and shareware, discussion forums, computing news, product and service reviews, and industry analysis.

The Cobb Group Online will offer the best of our journals, as well as authoritative articles and columns from Ziff-Davis publications, such as *PC Magazine* and *PC/Computing*. We'll also offer you the best shareware and public-domain utilities.

ZiffNet services cost \$2.50 per month plus connect-time charges of \$12.80 per hour for 1200 or 2400 BPS or \$22.80 per hour for 9600 BPS. The following information will help those of you who have modems to log onto ZiffNet. (If you're new to going online, we've also provided a brief glossary to help you get started.)

Logging onto ZiffNet

You can download the batch files that appear in this issue from ZiffNet. There are two procedures for accessing ZiffNet, depending on whether you already have a CompuServe ID.

If you're a member of CompuServe

If you're already a CompuServe subscriber, just type `GO ZNT:COBB` from any CompuServe ! prompt.

If you're not a member of CompuServe

If you're not a member of CompuServe, you must get a ZiffNet ID. To do this, first determine your local access number by calling (800) 635-6225 or, if you have a touch-tone phone, you can select the options for access information, local access numbers, your country, and then type in your area code and telephone number. (Calling a local phone number means you won't have to pay any long-distance charges to access ZiffNet.) Then, you'll sign onto ZiffNet and enter your billing information. After you do, the system will assign you a ZiffNet ID and a password.

To get your ZiffNet ID

Once you have your local access number, you're ready to connect to ZiffNet. Set your communications software to 7 data bits, even parity, one stop bit. Select a transfer rate of 1200, 2400, or 9600. Then, have your modem dial your local access number. After you're connected, respond to the prompts by entering the following:

When you connect: [Ctrl]C
Host Name: CIS
User ID: 177000, 5555
User Password: ZIFF*NET
Enter Agreement Number: COBBAPP

Now, register your name and credit card number for billing purposes. Make sure you write down your user ID and password. You'll need them to log on for the next ten days. Within this time, you'll receive by mail a replacement password to confirm your membership. ■

Online glossary

You can bring a vast amount of information to your desktop if you have a few key pieces of hardware and software and if you're a member of an online service. Here are the definitions of some key terms you'll probably run into:

- **baud:** How many bits a modem can transmit per second. Modems with high baud rates usually cost more, but they might save you charges for connect time.
- **connect time:** The time you spend linked to an online service. Many services charge for each minute you're connected, much like long-distance telephone services.
- **communications software:** A special program that lets your computer control and communicate with your modem. Many modems include a communications software package, such as ProComm from DataStorm technologies or Smartcom from Hayes.
- **database:** A collection of information that you can search. For example, you might search a database of magazine articles to look for reviews of computers you're thinking about buying.
- **forum:** A service that allows you to exchange information about a particular topic. For example, The Cobb Group Online will allow you to post messages about DOS. You can also read and respond to messages from other DOS users.
- **log on:** To connect your computer to an online service. You'll usually need to enter a user ID and password to log on.
- **modem:** A device that lets your computer use the telephone line to communicate with another computer.
- **online service:** A business that lets you access information by dialing into its computer.

Selecting larger display type with ANSI codes

If you've ever left a long session on your PC with bleary eyes, you know that looking at small, glowing type can wear you down after a while. Although many applications offer a choice of soothing color schemes and display options, most users simply accept the default display options when they're working at the DOS command line.

As you may have read in "Using the PROMPT Command to Color Your DOS Screen" in the May 1992 issue of *Inside DOS*, the ANSI.SYS driver allows users with color monitors to select foreground (type) and background colors. While selecting a soothing color combination can help prevent eye fatigue, you're still left with the problem of straining to read characters in the default type size, about .25 inches tall. Fortunately, as we'll show you in this article, the ANSI.SYS driver will allow most DOS users to select a larger type size for their displays as well. You can create the batch file described in "Choosing Display Modes with SCREEN.BAT" on page 6, which will let you easily switch display options. First, let's get started by reviewing the file that makes all of this possible: the ANSI.SYS driver.

ANSI.SYS basics

As you may recall from the article in the May issue, ANSI.SYS is a device driver you install through your CONFIG.SYS file. ANSI.SYS doesn't do anything by itself, but it does allow you to use special commands to control various aspects of your keyboard and display. To be able to use these commands, you must place the following line in your CONFIG.SYS file:

```
device=c:\dos\ansi.sys
```

(assuming that the ANSI.SYS file is in your C:\DOS directory). If you're using the EMM386.EXE driver or another UMB provider and you've also specified *dos=umb* in your CONFIG.SYS file, you can use DEVICEHIGH instead of DEVICE. Loading ANSI.SYS with DEVICEHIGH will leave you with a little more conventional memory for running other programs.

Entering ANSI codes

Once you've loaded the ANSI.SYS driver, you can issue ANSI commands from the DOS prompt or through a batch file. Either way, you'll need to tell DOS you're using ANSI.SYS commands by placing a special series of characters, called an escape sequence, in front of the codes that control your monitor or keyboard. The escape sequence consists of the escape character followed by a bracket.

Entering the escape character—also known as ASCII 27 or [Ctrl][—is a bit tricky. If you use the PROMPT command to enter an ANSI escape sequence, you represent the escape character by typing *\$e*. Then

you type an opening bracket ([) to complete the escape sequence and introduce the ANSI code. You'll need to use the PROMPT method to enter ANSI codes from the command line and immediately see their results. You also can use the PROMPT command within a batch file to enter ANSI codes.

Another way to enter ANSI commands is to echo them to the monitor. We use this method in "Choosing Display Modes with SCREEN.BAT." Unfortunately, you can't use the easy-to-remember *\$e* symbol to enter the escape character when you use the ECHO command. How you enter the escape character depends on the word processor you're using. In the DOS 5 Editor, you press [Ctrl]P, then [Ctrl][to create the escape character. In many other word processors, you hold down the [Alt] key and type 27, usually on the numeric keypad. The DOS 5 Editor and most word processors represent the escape character as a small left arrow (←).

Choosing display modes

Now that we've reviewed the basics of ANSI.SYS, we're ready to look at the display mode options, which will allow you to select different type sizes for your DOS screen. If you have a VGA, EGA, or CGA monitor, you can select from the seven display modes shown in Table A. We've included both the codes you can type from the command line when you use the PROMPT command, as well as the codes as they would appear in the DOS 5 Editor if you echo them in a batch file.

Table A

Prompt	Echo	Effect
<i>\$e</i> [=0h	←[=0h	40-by-25 monochrome text
<i>\$e</i> [=1h	←[=1h	40-by-25 color text
<i>\$e</i> [=2h	←[=2h	80-by-25 monochrome text
<i>\$e</i> [=3h	←[=3h	80-by-25 color text
<i>\$e</i> [=4h	←[=4h	320-by-200 four-color graphics
<i>\$e</i> [=5h	←[=5h	320-by-200 monochrome graphics
<i>\$e</i> [=6h	←[=6h	640-by-200 monochrome graphics

You enter the codes for screen modes slightly differently, depending on if you can enter them at the prompt or through the ECHO command.

The effects of the modes vary, depending on the display you have. You'll probably want to experiment with the codes to see which mode is your favorite. Of course, if you have a monochrome monitor, you won't see the effects of the color codes.

To give you a better idea of what these settings do, we've included some sample screens using the modes on a VGA monitor. Figure A shows an example of the 40-by-25 monochrome text mode, which seems

wider and a little squat. You could choose the 40-by-25 display by typing

```
C:\>prompt $E[=0h
```

on the command line and pressing [Enter]. Figure B shows the 80-by-25 monochrome text mode, which is your system's standard mode. You can choose it from the command line by entering

```
C:\>prompt $E[=2h
```

Or, for 80-by-25 color text, enter

```
C:\>prompt $E[=3h
```

These prompts will produce your monitors default display mode. On the other hand, Figure C shows large, rougher typeface produced by the 320-by-200 graphics modes. If you have a color VGA monitor and enter the code

```
C:\>prompt $E[=4h
```

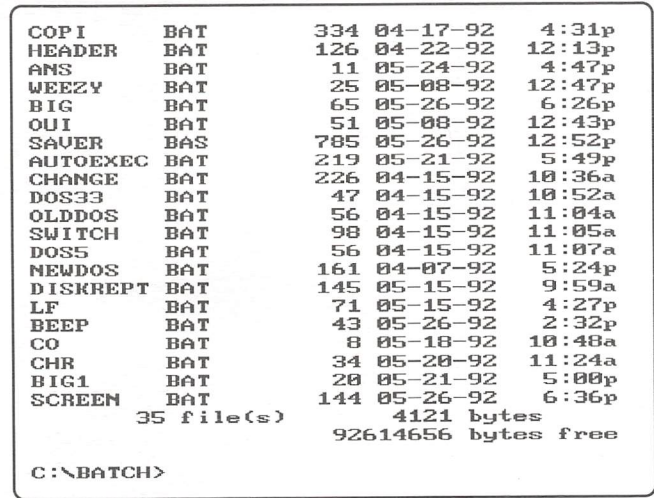
you might see some strange effects if you normally run DOS with an ANSI screen color. That's because the new prompt will use a black background and translate your foreground (type) color, as shown in Table B.

Table B

Codes	Colors	Translated color
30, 31	Black, Red	Black
32, 33	Green, Yellow	Purple
34, 35	Blue, Magenta	Cyan
36, 37	Cyan, White	White

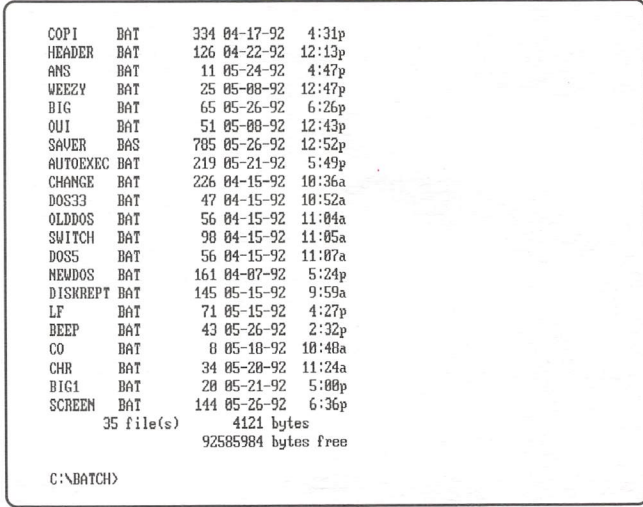
Color VGA monitors may change the colors you've selected to these four when you use a four-color mode.

Figure A



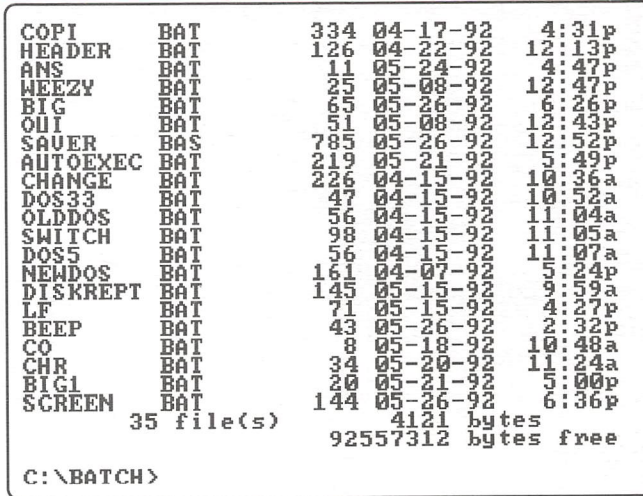
Text on your DOS screen looks bigger and wider in 40-by-25 mode on a VGA monitor.

Figure B



The 80-by-25 mode is the default for most monitors.

Figure C



Text looks largest in 320-by-200 mode, but it also has rough edges.

Stabilizing the display

When you enter these prompts, you'll probably notice that they clear the screen of everything: your system prompt, as well as any previous commands you issued. Furthermore, when you try to issue another command, such as DIR, DOS will flash the output of the command, then clear the screen. Obviously, you won't get much useful information this way! However, you can prevent DOS from clearing the screen after each command by resetting the prompt you want to appear on the command line. A common prompt, for example, is the path and greater than sign, which you enter with the codes \$p\$, as shown below:

```
C:\>prompt $p$
```

Notes

Even though you can stabilize the display by resetting your system prompt, you still might notice some strange effects after changing display modes. The 320-by-200 mode shown in Figure C is most likely to produce strange effects. We've already shown you, in Table B, how the four-color mode may translate colors you've chosen. Some applications also may look strange. For example, if you choose 320-by-200, four-color mode (4h) and run EDIT.COM, the DOS 5 Editor screen will be cyan and yellow, instead of its usual dark blue and gray. Also, you'll need to clear the screen by issuing the CLS command when you exit from the Editor. If you don't

clear the screen, any new command you type will overwrite commands issued before you ran EDIT. If you discover side effects with other programs, you can use SCREEN.BAT, described in the following article, to select the normal display mode before you run the program.

For more information on controlling your display with ANSI.SYS codes, see the following articles:

"Using the PROMPT Command to Color Your DOS Screen," May 1992

"Adding a Header Line to Your DOS Screen," June 1992

For information on ordering back issues, see the masthead on page 2 of this issue.

BATCH FILE

Choosing display modes with SCREEN.BAT

In the article "Selecting Larger Display Type with ANSI Codes" on page 4, we showed you some ANSI codes that change display modes. We also showed you how to experiment with the modes by entering them with the PROMPT command. If you followed along, you probably found the modes that work best for you. Now, we'll show you how to create a batch file that will switch between the normal display mode and the large-type mode that works best for you.

Let's suppose, for example, that you like the 40-by-25 color mode (1h), but you also sometimes like to use the default 80-by-25 mode (3h). SCREEN.BAT, shown in Figure A, allows you to switch between the two modes by typing *screen* followed by the "name" BIG or NORM. (We've used only two settings in SCREEN.BAT to keep the file simple. You can add as many settings as you'd like by creating new labels and adding the necessary prompts after each one.)

Figure A

```
@echo off
goto %1
:BIG
echo ← [=1h
goto TIDYUP
:NORM
echo ← [=3h
goto TIDYUP
:TIDYUP
prompt $p$g
cls
```

SCREEN.BAT lets you change display modes quickly.

As you can see, SCREEN.BAT uses labels so that it can contain different sets of instructions. Let's take a closer

look at how the batch file works. The first line, *@echo off*, simply tells DOS not to display each command as it executes it. The big advantage of turning off echo is that it helps prevent those annoying double-prompts that can show up after you run a batch file. The next line,

goto %1

tells DOS to look for the parameter you type after *screen* when you run the batch file. Specifically, the %1 symbol tells DOS to look for a label in the batch file that matches the first word you typed right after the SCREEN command. When it finds the label, DOS will execute the commands in that section. As we said earlier, the choices are BIG and NORM.

The next section of SCREEN.BAT begins with the :BIG label (remember to include the colon before the label name):

```
:BIG
echo ← [=1h
goto TIDYUP
```

The line after the :BIG label echoes the ANSI code for the 40-by-25 color text display. In the DOS 5 Editor, you enter the ANSI code

← [=1h

by pressing [Ctrl]P, then [Ctrl][to create the escape character (←). Then you type another opening bracket, the equal sign, the number 1, and a lowercase h. Next, the :BIG section ends by sending you to the :TIDYUP label.

As you might expect, the :NORM section that follows echoes the ANSI code for the default 80-by-25 color display:

```
echo ← [=3h
```

Like the :BIG section, the last command under the :NORM label goes to the :TIDYUP section.

The last section of the batch file, which begins with the :TIDYUP label, is shown below:

```
:TIDYUP
prompt $p$g
cls
```

Because the :BIG and :NORM sections both clear the system prompt, you need the :TIDYUP section to reset the prompt. To keep things neat, we've also included the CLS command to clear the screen.

Using SCREEN.BAT

You can specify which mode you want by using a DOS parameter when you run the batch file. To do this, you simply type the name of the file (you can leave off the BAT extension), followed by the name for the setting you want to use. When you run a batch file, DOS interprets the command like this:

```
screen %1
```

So if you type

```
C:\>screen big
```

DOS will substitute the word *big* wherever it sees the %1 symbol used in SCREEN.BAT. By doing so, it will go to

the :BIG label and echo the ANSI code that selects the 40-by-25 color text display mode.

You can reset the screen to its display mode just as easily. To do this, you simply enter the command

```
C:\>screen norm
```

Of course, if you don't type a valid label name, DOS won't be able to carry out the change you intended. So, if you don't type *big* or *norm* after *screen* when you run SCREEN.BAT, DOS will quit the batch file and present the message *Label not found*.

Notes

Since SCREEN.BAT uses the ECHO command to set the screen mode, you'll have a surprise if you ever try to issue the command *type screen.bat*. When it encounters the ECHO command followed by a code in the batch file, DOS will send the code directly to the monitor. You'll see the mode change a few times and settle on whichever mode is last in the file—in this case, the :NORM setting. ■

For more information on the batch file technique, see the following article:

"Making Your Batch Files More Flexible with GOTO,"
October 1991

For information on ordering back issues, see the masthead on page 2 of this issue.

VAN WOLVERTON

Taking care of the environment

Part of the frustration with using a computer is trying to keep track of the jargon. Admittedly, new technology often requires new terminology, but computer folks seem to go overboard sometimes, especially when it comes to applying different meanings to familiar terms. Booting the system, for example, has nothing to do with kicking it or getting rid of it (although occasionally we may be tempted...).

So it is with the DOS environment, which has nothing to do with trees or streams or bunnies. This environment is something much more mundane: an area of memory that DOS reserves for storing information that remains unchanged, regardless of what program is running, unless you or a program explicitly change the information. Information stored in the environment is available to any program that runs under DOS.

Two items of information stored in the environment are familiar to most of us: the definition of the command prompt and the names of the directories that make up

the command path. As we'll see in this article, you also can use the environment to store variable values that will make your batch files more flexible.

Environment variables

DOS stores information in the environment in the form of a name and a value separated by an equal sign (for example, USER=Fred). This combination is called an *environment variable*. Except for the path and prompt definitions, DOS itself pays no attention to environment variables; they're for the programs to use.

It's easy to see what variables are stored in your DOS environment—just type *set*, then press [Enter]. DOS should display something similar to the following lines:

```
PATH=c:\batch;c:\dos;c:\word;c:\win31;c:\ipm
PROMPT=$p$g
TEMP=c:\win31\temp
USER=Fred
```

Although your environment will contain different variables, some should look much like this example. The first two environment variables (PATH and PROMPT) define the command path and system prompt. Chances are, your environment includes variables named PATH and PROMPT as well, although their values probably differ from ours. Your path and prompt lines might be in a different order. Your path will almost certainly be different, since it should include directories that contain the commands and programs you use most often. But you're more likely to have the same prompt as ours, since we're using the common \$p\$g prompt. This prompt will display the drive letter and path (represented by the code, \$p), followed by the greater than sign (\$g). For example, if you're in the C:\DOS directory, the \$p\$g prompt will display C:\DOS>.

So far, we've shown you how the SET command lets you display the variables stored in the environment. Now, let's look at how the SET command lets you create or delete your own environment variable.

Creating an environment variable

When you define an environment variable with the SET command, DOS converts the name to uppercase letters, regardless of how you enter it, but leaves the value just as you type it (uppercase and lowercase letters represent different values). For instance, both the following commands would create an environment variable like the last one in our example on page 7:

```
C:\>set user=Fred
C:\>set USER=Fred
```

Some programs create one or more environment variables by putting a SET command in your AUTOEXEC.BAT file when you install them. For example, the third variable (TEMP) in the preceding sample was created by the command SET TEMP=C:\WIN31\TEMP, which the Windows 3.1 installation program placed in our AUTOEXEC.BAT file.

You also can use SET to delete an environment variable by following the name only with an equal sign. For example, if you want to delete the environment variable USER=Fred, you type

```
C:\>set user=
```

Special cases: PATH and PROMPT

Of course, there are a few exceptions to these tidy rules. DOS is sprinkled with the occasional maddening inconsistency, such as this: You can define the command path and prompt environment variables not only with the

SET command, but also with the PATH and PROMPT commands. However, the rules are a little bit different depending on which command you use. For example, the following commands all set the command path to the directories C:\DOS and C:\WORD:

```
C:\>set path=c:\dos;c:\word
C:\>path=c:\dos;c:\word
C:\>path c:\dos;c:\word
```

Notice that the SET command requires you to enter an equal sign between the name and value of the environment variable, but PATH works properly whether or not you include an equal sign. The same is true when you use the SET or PROMPT commands to define the system prompt.

It gets a little messier when you delete a value. Both of the following commands delete the path:

```
C:\>set path=
C:\>path=;
```

However, if you type *set path=;*, DOS sets the command path to a semicolon (which is useless). If you type *path=* or just *path*, DOS displays the directories in the command path and doesn't change the environment. The alternatives multiply when you want to delete the prompt definition. Any of these commands will do the trick:

```
C:\>set prompt=
C:\>prompt
C:\>prompt=
C:\>prompt=;
```

Again, if you type *set prompt=;*, DOS sets the prompt to a semicolon. When you delete the PROMPT variable, DOS reverts to its standard prompt (C>).

Environment variables in batch files

You've probably used replaceable parameters in a batch file. When you write the batch file, you use the percent sign followed by a single-digit number to represent the value of the parameter. You enter the values you want to use for the parameter after the batch file name when you run it. DOS replaces %1 with the first parameter typed with the batch command, %2 with the second parameter, and so forth. (The article "Choosing Display Modes with SCREEN.BAT," which begins on page 6, shows you how to create a batch file that uses a replaceable parameter.)

You can use environment variables in a similar manner. When DOS carries out a batch file command that includes the name of an environment variable

preceded and followed by a percent sign, it replaces the name of the environment variable with the corresponding value. For example, the following batch file displays the names of each directory in the command path, separated by semicolons (just as you would enter them in the PATH or SET commands):

```
@echo off
echo %path%
```

DOS replaces *%path%* with everything to the right of the equal sign in the PATH environment variable and displays it.

You also may have used the FOR batch command to carry out another command once for each file whose name matches a filename with wildcards. FOR behaves the same way with an environment variable whose value, like PATH, consists of several values separated by a semicolon. When you want to use a variable that can represent multiple values, you assign it a name consisting of two percent signs, followed by a letter (*%a*, *%b*, and so forth). For example, you can create the LISTPATH.BAT batch file shown below to display the name of each directory in the command path:

```
@echo off
echo Directories in Command Path
for %a in (%path%) do echo %a
```

For each directory name separated by semicolons, the FOR command carries out the ECHO command that displays the directory name. If your path is set as

```
PATH=c:\dos;c:\batch;c:\word
```

LISTPATH.BAT will display the following result:

```
c:\dos
c:\batch
c:\word
```

The technique of using a variable to represent multiple values works not only when the individual values are separated by a semicolon, but also when you use a space, tab, comma, period, or slash. (In our example, however, the directory names that make up the value of the PATH environment variable must be separated by a semicolon for DOS to accept them.)

Saving your path and prompt

Experimenting with environment variables can delete either your command path or prompt definition. Since it's a pain to retype them if they're long, let's look at a couple of ways to create batch files that restore your current command path and prompt.

When you type *path* and press [Enter], DOS displays *PATH=* followed by the names of the directories in the command path, separated by semicolons; this happens to be the proper form for a PATH command. So, to create a

batch file that will restore your current command path, all you have to do is redirect the output of the PATH command to a file named PATHREST.BAT:

```
C:\>path > pathrest.bat
```

Now you can restore your command path simply by typing *pathrest*.

You have to do a little more work to create a batch file to restore your current prompt definition because there's no command that tells DOS to display that information. But the SET command displays all the environment variables, including the one named PROMPT. You can still capture the current prompt setting by piping the output of the SET command to a FIND command that searches for PROMPT, then redirecting the output of the FIND command to a file named PRMPTRST.BAT:

```
C:\>set | find "PROMPT" > prmptrst.bat
```

Once you've issued this command, you can restore your prompt by typing *prmptrst* and pressing [Enter]. After you create PATHREST.BAT and PRMPTRST.BAT, you can experiment to your heart's content without having to retype either a PATH or PROMPT command.

Adding a directory to the path

Sometimes you might like to add a directory to the command path temporarily, but you don't want to type the name of every other directory in the path. A simple batch file lets you add a directory to the end of the list of directories in the path. Create ADDPATH.BAT:

```
@echo off
path=%path%;%1
```

DOS replaces *%path%* in the second command with the current value of the PATH environment variable, which is your current command path. Immediately following *%path%* is a semicolon and *%1*. When you run the batch file, you can define *%1* by typing a directory name after *addpath*. When you do this, the second line of ADDPATH.BAT becomes a PATH command that specifies all the directories in the current path, plus a semicolon and the name of the directory to be added. For example, if your current path is defined as

```
PATH=c:\dos;c:\batch;c:\word
```

and you want to add the C:\EXCEL directory to it, you simply type the command

```
C:\>addpath c:\excel
```

After you press [Enter], ADDPATH.BAT will reset your path to

```
PATH=c:\dos;c:\batch;c:\word;c:\excel
```

Remember, this new path remains in effect only until you enter another PATH command or turn off your system. If you define your path in AUTOEXEC.BAT and want to add the directory permanently, you'll have to edit AUTOEXEC.BAT and add the new directory to the PATH command there.

Changing the environment size

If you don't specify otherwise, DOS sets aside 160 bytes for the environment. You can quickly use up that space if you include several path names in your command path or define an elegant prompt.

If you try to assign an environment variable that would exceed this space, DOS assigns as much of the variable name and value as will fit in the remaining space, then replies *Out of environment space*. This results in a partial assignment, which almost always is wrong. Usually, you'll have to delete the variable that's causing the overflow.

Fortunately, you often can prevent overflows by telling DOS to set aside more room for the environment. You do this by placing in the CONFIG.SYS file a SHELL command that includes the /E (for *environment*) parameter. After the /E parameter, you type a colon,

followed by the number of bytes you want to set aside for the environment. As a final step, you type the /P parameter, which tells DOS to make the command interpreter permanent. For example, to tell DOS to set aside 512 bytes for the environment, you would include the following SHELL command in CONFIG.SYS (assuming that COMMAND.COM is the command interpreter and it is stored in the directory named DOS in the root directory of drive C:):

```
shell=c:\dos\command.com /e:512 /p
```

This SHELL command should be enough, unless you have an exceptionally long command path and prompt definition or unless many programs have created environment variables for their use. If you find that you still get the *Out of environment space* message, increase the size even more. ■

For more information on environment variables, see the following articles:

"Fun with the DOS Prompt," March 1992

"Taking a Walk on the DOS Path," May 1991

For information on ordering back issues, see the masthead on page 2 of this issue.

Continued from page 1

Strategies for organizing directories

screen with the directory heading information that the DIR command displays.

Of course, you shouldn't sacrifice the logic of your organizational scheme just to ensure that your directory listings don't extend beyond one screen. After all, the command *dir/p* will pause the display after the directory fills a screen, while *dir/w* will display five columns of file names on the screen.

Programs vs. data

So far, we've focused on the number of files you should have in a directory. A more meaningful principle for organizing directories is to separate program files from data files. A *program* file is any file that carries out a command, runs a program, or supports a program. When you install software, the installation program will usually create a directory and place all the program files in it. For example, installing Microsoft Word creates a C:\WORD directory. The installation program then copies the files you need into that directory.

Some DOS users simply place all the files they create in the application's directory. After a while, however, they end up with a mishmash of data files and program files. Having so many diverse files in one directory can make it difficult to locate the files

you want to open from within the program. Furthermore, when you use the asterisk wildcard (*) to delete files from the program directory, you risk accidentally deleting a program file with a similar name. Of course, deleting a program file would disable the software until you reinstall it.

Keeping data files in a directory separate from your program files not only helps you avoid accidents, but the practice also can help you back up data more efficiently (depending on the method you use). Since you already own the installation diskettes for your program and you've already made backup copies of them (we hope), you don't really need to back up your program files every time you back up your data. If you separate data files from program files, you need only back up your data directories routinely. On the other hand, if you store data files in your program directory, you'll waste time and disk space creating redundant copies of your program files.

Categorizing information

Limiting the number of files in your directories and separating data files from program files are fairly straightforward techniques for organizing your directories. But logically categorizing information is a much

more abstract—and personal—topic. Let's take a look at the advantages and disadvantages of organizing by subject or by application.

Organizing by subject

Perhaps the most intuitive way to manage information is to organize it by subject. In fact, you use this method when you label folders for your filing cabinet. You might have folders titled *Health Insurance*, *Expense Reports*, *Wilson Project*, or *Building Codes*.

Some folders, such as the one labeled Wilson Project, may contain many different types of documents, although these are unified by the folder title. For example, the Wilson Project folder may contain a proposal, a surveyor's report, bids from contractors, a specifications list, a spreadsheet with projected costs, a schedule, a petition to the zoning commission, as well as numerous letters and memos.

You could create a directory on your hard drive to contain the same information. For example, suppose you created a directory named C:\WILSON. You could then save the budget spreadsheet you created in Excel as C:\WILSON\WIL_BUDG.XLS. If you created the specifications list in Word, you could save it as C:\WILSON\WIL_SPECS.DOC and the letter to the zoning commission as C:\WILSON\WIL_ZONE.DOC.

Organizing by application

Without a doubt, organizing files by subject matter helps you locate the information you need. But organizing by subject has its drawbacks, too. A minor problem is that you might end up creating more directories than you would with another organization scheme. More important, you might find that you have to change directories frequently from within your applications. If you're using an application with an awkward file management system, you might spend a lot of time changing directories in order to load a particular data file. For example, suppose your default directory in Word is C:\WORD. You'd have to choose the C:\WILSON directory through the program's Open dialog box when you wanted to edit C:\WILSON\WIL_SPECS.DOC. And, unless you changed the default directory from within the application, Word would continue to use the C:\WORD directory as the default whenever you load files or save new files.

If you really hate changing directories from within an application, you can create a subdirectory in your application directory to store all the data files you create with the application. (The sidebar below shows you how to create a directory with the MD command.) Then, you can use that subdirectory as the program's default

A roundup of directory commands

In the article "Strategies for Organizing Directories" we show you some rules of thumb for deciding where to put files on your hard disk. If you've decided to reorganize your hard disk, here's a brief review of the commands you'll need.

Changing directories

You can change to a directory from any DOS prompt by typing `cd \` followed by the directory's path and name. (Of course, if you want to change to the root directory, you simply enter `cd \`.) For example, here's how you'd change from the C:\WORD directory to the C:\EXCEL directory:

```
C:\WORD>cd \excel
C:\EXCEL>_
```

To change to a subdirectory of the current directory, you just type `cd` followed by the subdirectory name:

```
C:\WORD>cd memos
C:\WORD\MEMOS>_
```

If you want to change to the parent of the current directory, you type `cd` followed by two periods, then press [Enter]. For example,

```
C:\WORD\MEMOS>cd..
C:\WORD>_
```

Making a directory

If you type `md` followed by a directory name, DOS will create a new subdirectory in the current directory. For example, to create the directory C:\WORD\LETTERS, you'd enter

```
C:\WORD>md letters
```

From a different directory, you'd enter the full path:

```
C:\EXCEL>md \word\letters
```

If you accidentally create a directory you don't need, you can remove it by typing `rd`, followed by its name:

```
C:\WORD>rd letters
```

directory. Returning to our example, you could place all your Word documents in the C:\WORD\DOCS directory. Although this technique is convenient in some ways, you might end up with an unwieldy number of files in the application's default directory. This is particularly a risk with word processing programs, since you may create several documents each day or week. You'd probably find creating a single data subdirectory more useful for database or spreadsheet applications.

The middle course

If neither organizing by subject nor organizing by application seems like a good choice for you, try compromising. Organize mainly by application, but allow for several subdirectories of data.

You still can use program directories as the basis for organizing your hard disk. Instead of storing files

you create in only one subdirectory of the program directory, create several categories for your data files. Although these categories can be based on content, you also could base them on file type. For example, you could make subdirectories for letters, memos, and reports in your word processing directory. In Word, you could name these directories C:\WORD\LETTERS, C:\WORD\MEMOS, and C:\WORD\REPORTS.

Conclusion

While organizing the directories on your hard disk is almost as individual as organizing your filing cabinet, we've presented a few rules of thumb for organizing your files. Limiting the number of files in a directory is a fairly straightforward rule, but choosing a logical way of categorizing your files is more complex and individual. ■

A directory of subdirectories

When you're managing your directories and deciding if you need to create new ones, you'll need to see what directories and subdirectories already exist. Most DOS users don't use an extension on their directory names, while they use extensions for all other types of files (BAT, WKS, or DOC, for example). If you use this convention, you can list the directories on your root directory by typing

```
C:\>dir *
```

DOS 5 provides another way to list directories independent of the naming convention. With DOS 5, you can view directories by typing

```
C:\>dir /a:d
```

then pressing [Enter]. The /A:D switch tells DOS to look for entries that contain the directory "attribute."

You can change to the root directory and add the /S switch to see all of the subdirectories on the hard drive, as shown below:

```
C:\>dir /a:d /s
```

If you have a lot of subdirectories, you'll probably want to add the /P switch to pause the listing after each screen. Or, you can direct the output of the command to the printer with the following command:

```
C:\>dir /a:d /s > prn
```

Finally, if you'd like a more graphical representation of your directories and subdirectories, you can issue the TREE command. Since the tree structure of your drive will probably fill more than one screen, you can pause the display by piping the command through the MORE filter with the following command:

```
C:\>tree | more
```

Or, if you prefer, you can enter

```
C:\>tree > prn
```

to send the directory tree to the printer. ■

